Autonomous Active Learning of Task-Relevant Features for Mobile Manipulation

Hai Nguyen, and Charles C. Kemp

Healthcare Robotics Lab, Georgia Institute of Technology, USA

***Abstract*—We present an active learning approach that enables a mobile manipulator to autonomously learn task-relevant features. For a given behavior, our system trains a Support Vector Machine (SVM) that predicts the 3D locations at which the behavior will succeed. This decision is made based on visual features that surround each 3D location. After a quick initialization by the user, the robot efficiently collects and labels positive and negative examples fully autonomously.**

To demonstrate the efficacy of our approach, we present results for behaviors that flip a light switch up and down, push the top or bottom of a rocker-type light switch, and open or close a drawer. Our implementation uses a Willow Garage PR2 robot. We show that our approach produces classifiers that predict the success of these behaviors. In addition, we show that the robot can continuously learn from its experience. In our initial evaluation of 6 behaviors with learned classifiers, each behavior succeeded in 5 out of 5 trials with at most one retry.

I. Introduction

Although mobile manipulators are becoming more capable, creating robust perceptual algorithms that tolerate real-world variation continues to be a challenge. One promising approach to this problem attempts to simplify perception in unstructured environments by identifying *task-relevant features* [6, 8]. In contrast to approaches that seek to infer and represent much of the world's state, methods that use task-relevant features recover only parameters that are needed by task-specific controllers.

Within this paper, we present an active learning approach that enables a mobile manipulator to autonomously learn task-relevant features. In this initial work, we consider the problem of reliably finding and recognizing positions on known mechanisms that predict success. Our approach makes several assumptions. First, we assume that the robot can execute a set of behaviors, each of which only requires a 3D location in the robot's frame of reference as initial input. We have previously demonstrated that this is a reasonable assumption for a variety of useful mobile manipulation behaviors [12]. Second, we assume that the robot has a way of reliably detecting whether or not a behavior it has executed was successful or not. Third, we assume that for each behavior, B, there is an inverse behavior, B^{-1}. If B successfully executes, then successful execution of B^{-1} will return the world to a state that allows B to execute again.

Given these three assumptions, our active learning method enables a mobile manipulator to autonomously learn which 3D locations are likely to result in successful execution of a behavior. More specifically, the robot learns to classify visual features associated with 3D locations as being associated with success or failure of a given behavior. By learning a direct mapping from perception to behaviors through experience, the robot automatically handles issues such as calibration errors, variations in the pose of the robot due to imprecise navigation, lighting changes and sensor noise. In contrast to many works in perception, our method is designed specifically to recognize locations on the same object across task variations instead of focusing on generalization across instances of the same object.

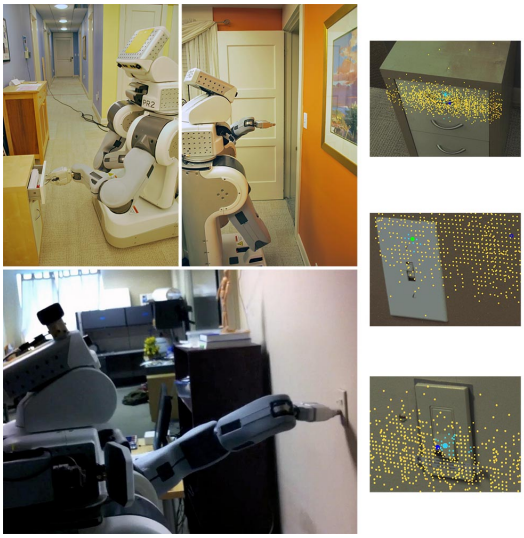

Fig. 1. **Left:** Willow Garage PR2 operating a drawer, light switch and rocker switch using learned task-relevant detectors. **Right:** Results from learned detectors during execution.

To construct an appropriate task-relevant detector for a particular behavior, our system uses a Support Vector Machine (SVM) to discriminate between the appearances of 3D points associated with success versus failure. We use active learning heuristic proposed by Shohn and Cohn to train an SVM for each behavior [16]. During training, this active learning method proposes which 3D point to experiment on next, which reduces the number of examples required. It also enables the robot to continue learning after training. The robot can adapt to situations that it has not encountered before by invoking the same active learning method. If the robot fails at runtime, it can hypothesize new likely locations, test and learn from its experience in the process.

To demonstrate the efficacy of our approach, we present results for behaviors that flip light switches up and down, push the top or bottom of a rocker-type light switch, and open or

close a drawer. Our implementation uses a Willow Garage PR2 robot. We show that our autonomous training system is able to produce classifiers that predict the success of these behaviors. In addition, we show that the robot can continuously learn from its experience. In our initial evaluation of six behaviors with learned classifiers, each behavior succeeded in 5 out of 5 trials with at most one retry.

II. Related Works

A. Task-Relevant Feature Detection

Recently, there has been recognition in the mobile manipulation community of the importance of exploiting task structure to reduce the complexity of operating in the real-world. This point was argued by Katz et al in [7]. Dang and Allen [2] show evidence that many manipulation tasks are well approximated using sequences of rotations and translations. Additionally, work in articulated object perception [6], tool tip detection [8], door handle detection [9], behavior-based grasping [5], and corner detection for towel folding [11] demonstrates that in many tasks a small set of simple features are sufficient to support object manipulation. Unlike our approach, these methods required hand-coded and hand-trained task-relevant feature detectors.

B. Learning Methods in Manipulaion

Even though the use of learning-based methods can result in more general detectors, labeled training examples are often time consuming and expensive to obtain. To get around this problem some researchers have used data from the web or simulation [9, 15]. Alternatively, systems can use data generated from self-experience [14, 3, 13, 10], which is the approach we have taken. A challenge in this setting is the cost of obtaining labeled examples, which we overcome by having autonomous methods of detecting success and failure for each behavior.

The work of Sukhoy and Stoytchev [17] is especially notable for its similarity to our approach. They have presented a system that uses an uncertainty sampling scheme to actively learn the appearance of door bell buttons. We extend that class of work with an approach that uses a different active learning algorithm, works with a mobile manipulator operating in situ devices, and can use more general behaviors that result in persistent changes to the state of the world.

III. Approach

We pair an SVM classifier with a multiscale PCA derived feature representation. The SVM classifier gives us a simple criteria for active learning, is resistant to overfitting, and is fast to evaluate. We pick the multi-scale PCA descriptor here as it is fast to calculate, data derived and does not make strong assumptions about scene contents.

A. Autonomous Training

1) Initialization: Our initializtion method is motivated by the idea that a person could take the robot on a tour and point out task-relevant locations using a laser pointer while specifying relevant behaviors. The robot could then autonomously navigate back to these locations and learn to robustly perform the behaviors through practice.

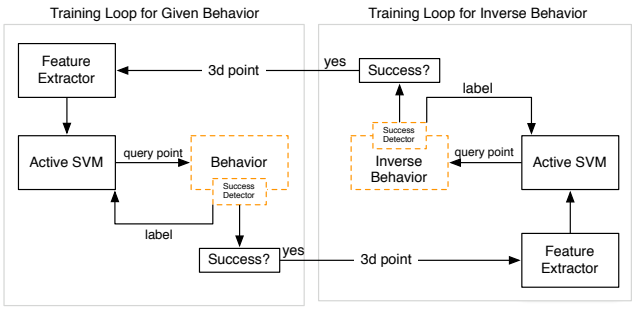

Fig. 2. Illustration of the classifier co-training procedure where we train the inverse behavior upon success of the primary behavior and vice versa. Dashed orange boxes on the two behaviors and success detectors highlight that these modules are provided as input to our system.

The process itself associates locations within an occupancy grid map with behaviors that can be applied, sets a target base position, and seeds the SVM learner with two starting examples, one positive and one negative. We first position the robot in front of the mechanism to be operated. Using a green laser pointer [12], we then provide an initial rough estimate for each behavior of a good 3D location. The robot samples 3D points around this designated location and executes the specified behavior at these points continuing until the robot gathers at least one positive and one negative point.

2) Training Procedure: During execution, when our robot navigates to the target base location stored in its map, there will likely be errors introduced in the relative position of the robot with respect to the mechanism. To capture this effect in our dataset, we designed our training procedure to sample from this distribution of positioning errors.

For our training procedure, we instruct the user to manually move the robot to four additional locations in the map a small distance away from the location collected during initialization. Then, at each step, our training procedure navigates the robot to one of these preset practice locations, then back to the mechanism location to empirically sample a new view.

After sampling a view, we use the SVM learning algorithm to repeatedly query for labeled data points. We stop querying either when we gather a maximum of 6 points from the view or once the active learning process converges. We impose this conservative maximum limit to prevent the learning heuristic from converging early as points in a given view are highly correlated to each other in their appearance. This process then runs until all practice locations converge. Where convergence for a location is defined as the iteration when we first sample a view where our active learning heuristic immediately declares convergence.

We illustrate this active SVM learning process in Figure 2. Notably, in the training of each behavior's classifier we also need to gather data for its inverse as without this co-training process, attempts to reverse the state of the world using a naive procedure can dominate the time required for learning.

3) Behavior Execution Procedure: After the above process converges, we now have a trained system that can detect task-relevant points. Our procedure begins with feature extraction then loops until the robot is successful at the given behavior. At each iteration, the robot first classifies points predicted to be successful. Then, using a kernel density estimator, the robot selects the mode of this cloud of points predicted to be successful in the 2D image. If execution fails, the robot adds the failing data point into the current training set, retrains the classifier, and repeats the procedure. Here, in contrast to systems where the execution process is independent of data gathering and training, the robot has the opportunity to improve its classifier if errors are made during execution.

B. Classification

Our classification task in this work is to distinguish between 3D points that lead to success and those that lead to failure given that 3D point's associated appearance information. As is standard in supervised classification, given a dataset of labeled examples $D = \{(x_1, y_1), ...(x_N, y_N)\}$, with $x_i \in \mathbb{R}^M$ representing the feature vector and $y_i \in \{-1, 1\}$ where -1 and 1 lead, respectively, to success and failure, we want to be able predict y_j for $(x_j, y_j) \notin D$. In our problem, each x_i contains local 2D appearance information associated with a candidate 3D point. To solve this classification task, we follow a discriminative approach using an SVM.

In our setting, we will typically have an unbalanced dataset problem with our dataset containing many more negative than positive examples. Since the SVM misclassification cost term is defined over all samples, it becomes possible to minimize the objective by trivially misclassifying all the positive samples. To prevent this issue, we use an SVM formulation which separates misclassification costs from the positive and negative class [1]:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C^+ \sum_{y_i=1} \xi_i + C^- \sum_{y_i=-1} \xi_i \\ s.t. \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{1}$$

Where $\phi(\mathbf{x})$ represents the standard SVM kernel function. In the above, the normal SVM misclassification cost constant C has been separated into C^+ and C^- which are, respectively, costs due to negative and positive misclassifications. In this work we set C^- to be 1, and C^+ to be the number of negative examples over the number of positive examples.

1) Active Learning Heuristic: At each step i of our training procedure, our goal is to select from a pool of unlabeled 3D points a point for our robot to label by executing its corresponding behavior. To do so efficiently, as suggested by Schohn and Cohn [16], we select the point closest to the current decision boundary provided that it is closer than the currently selected support vectors. In [16], the authors showed empirically that this combination of stopping criteria and heuristic often allows the resulting SVM to obtain better performance than using the entire dataset itself while using fewer examples.

Using this heuristic, at each step i, we define the previous iteration's dataset as D_{i-1}, the set of support vectors as $X^{sv}_{i-1} = \{x^{sv}_1, \ldots, x^{sv}_P\}$, the pool of possible query points as $X^q_i = \{x^q_1, \ldots, x^q_M\}$, and the SVM distance function, which measures distance to the decision boundary, with $d(\mathbf{x_i}) = \left|\mathbf{w}^T\phi(\mathbf{x}_i) + b\right|$, then pick the closest point to the decision boundary from the set of query points that are all closer to the decision boundary than the support vectors using:

$$\underset{\mathbf{x}^q_i : \forall \mathbf{x}^{sv}_j \; d(\mathbf{x}^q_i) < d(\mathbf{x}^{sv}_j)}{\operatorname{argmin}} \quad d(\mathbf{x}^q_i) \tag{2}$$

2) Features: We extract 2D features for classification by using a color high resolution image, a point cloud, and a reference 3D point $\bar{\mathbf{p}} \in \mathbb{R}^3$. To capture local 2D appearance, we take the 2D projections into the camera of a point from the point cloud, $proj(\mathbf{p}^c_i)$, and capture square windows of pixels of successively increasing size centered at the 2D point. These windows are then scaled down to a fixed size number of pixels. By using this image pyramid representation, we capture the point's local context at multiple scales. We then use Principle Components Analysis (PCA) to project the set of image windows down to manageable number of coefficients.

As it is computationally expensive to search every point in the entire point cloud, and we want to constrain the area being classified using a prior distribution we define a Gaussian distribution $\mathcal{N}(\bar{p}, \Sigma)$ to guide our search process. For this distribution $\Sigma = diag(v_x, v_y, v_z)$ with v_x, v_y, and v_z being, respectively, variances in the x, y, and z direction. The Gaussian mean, $\bar{\mathbf{p}}$, is set as the mean of known past successes. We then extract features at points sampled without replacement from the captured point cloud according to the PDF of this distribution.

IV. Implementation

A. Learner Parameters

Our sensor processing pipeline begins with capture of a 3D point cloud from the PR2's narrow angle projected stereo, and a 5 megapixel color image from the Prosilica camera. We then sample points to extract features as described in Section III-B2. For each 2D projection of a 3D point sampled we capture a series of local image regions from the color high resolution image with square window sizes of 41, 81, 161, and 321 pixels. Each window, in turn, is then scaled down to a square window of size 31x31 pixels creating an RGB image strip of size 31x124 pixels for each data point. To reduce this 11532 valued vector to a manageable size, we calculate a set of 50 PCA basis vectors, for each action, using points sampled from the first image taken during initialization.

For classification of the above feature vector, we use an SVM with a radial basis function kernel. To set the hyperparameters of this kernel we used grid search with a training/testing set split on a labeled training set obtained through human labeled feature points extracted from a scan of a light switch.

TABLE I
TRAINING EXAMPLES GATHERED GATHERED FOR EACH ACTION.

Action	Positive Ex.	Negative Ex.	Total
Switch On	17	43	60
Switch Off	20	31	51
Drawer Open	23	35	58
Drawer Close	23	39	62
Rocker On	49	96	145
Rocker Off	47	94	141

TABLE II
NUMBER OF TRIES UNTIL SUCCESS (TRIED 5 TIMES PER ACTION).

Action	1^{st} Try	2^{nd} Try
Switch On	3	2
Switch Off	5	
Rocker On	2	3
Rocker Off	4	1
Drawer Open	5	
Drawer Close	5	

B. Light Switch Behaviors

Our light switch behavior is a series of Cartesian movement commands using low stiffness settings where the robot first reaches to the target point until the end effector makes contact. After contact, depending on whether the goal is to turn the lights on or off, it moves downward or upward, stopping after detecting a spike in acceleration.

While moving, our procedure detects success by taking the difference of the average intensity between a frame captured prior to moving and another frame captured after moving. As the light switch is fairly simple, its inverse behavior is identical except for a change in direction of pressing.

C. Rocker Switch Behaviors

Our rocker switch procedure consists solely of a reaching out step similar to the first step of the light switch behavior above. As the force applied from contact during the reach procedure is enough to activate the switch, it is the only action that we need. This behavior also uses the same image differencing mechanism for detecting success.

D. Drawer Behaviors

Pulling open and pushing closed a drawer requires slightly different behaviors and success detection mechanisms. Our pulling behavior first reaches to the drawer handle location, moves back slightly, grasps (with the same reactive grasper as in [4]), then pulls. For pulling, we define failure both as failure to grasp the handle and failure to pull, while remaining in contact, for at least a threshold distance (10 cm). With pushing, we define failure also as failure to acquire contact and remain in contact with the drawer surface for a certain threshold distance. Intuitively, this criteria classifies events where the robot pushes against a closed drawer, or immovable parts of the environment as failed cases.

V. EVALUATION

Prior to training we create an occupancy grid map using the PR2's navigation stack. We then evaluate our proposed method by training classifiers for the six separate actions described in the following sequence: switch lights off, switch lights on, drawer open, drawer close, rocker switch off, and rocker switch on.

After an initialization process (Section III-A1), we ran our active learning practice procedure (Section III-A2) until convergence. For each action at each location we seed the process with four locations in the room at places where our robot would likely travel from to get to the mechanism to be operated on. We allow this process to run mostly without interference, only pausing and resuming when our robot's batteries run low. A brief summary of the data captured through this process is presented in Table I.

To evaluate the classifier created by the above process, in each trial, we moved the robot to a random location in the room then started execution of the full mobile manipulation behavior being tested. We tested each action at each location 5 times for a total of 10 trials per locations as there are two behaviors defined per location. Within each trial we allowed for the behavior to retry and incorporate information from failures if it did not succeed the first time. However, since the behavior would perform better, we discarded the information learned within each trial each time we started a new trial.

We present results of our training process in Table I and Figure V, and summarize results of the execution process is summarized in Table II.

Results from Figure V shows that our method was able to identify the relevant regions to aim the light switch and rocker switch behaviors depending on whether the task is to turn the mechanism on or off. With the drawer handle, as grasping closer to the edge of the handle is less likely to be successful, our method correctly identified that the center is the better location to try to grasp.

However, during training of the rocker switch misclassification of whether the lights changed state or not caused our learning process to require at least twice as many samples until convergence as needed for the other training trials. Encouragingly, due to the built in retraining mechanism and knowledge from past trials, our execution process attained a 100% success rate (Table II) after at most two retries with each action.

VI. DISCUSSION AND CONCLUSIONS

We proposed a process which allows a mobile manipulator to autonomously create task-relevant feature detectors given a behavior and its inverse. Our system solves a common issue in behavior-based robotics where specifying the motions to accomplish a task is easy for programmers but creating a robust perceptual algorithm is not. Examples of problems in this class can be seen in the operations of all three mechanisms which we considered as each required only fairly simple controllers.

In addition, as a consequence of online learning and having a success classifier, our system is able to monitor future performance and learn from its own mistakes, a desirable trait for systems which operate in unstructured human environments. In

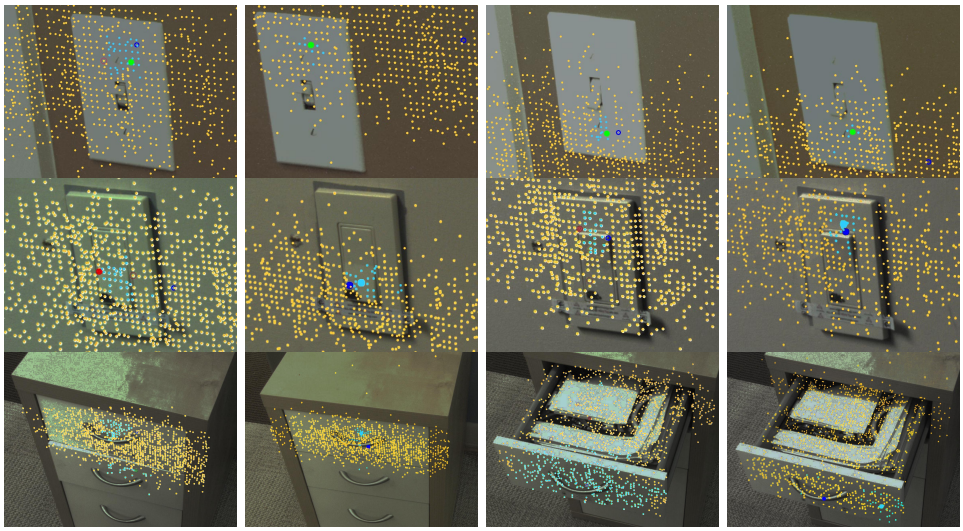

Fig. 3. Each pair of images shows results of learned detectors just after convergence and on a test scene. Orange points indicate locations where the behavior is predicted to fail, teal points indicate locations where the behavior is predicted to succeed, large teal points indicate locations in which the behavior will execute, and blue points are the mean of recorded successful locations determined through localization. **Row 1:** Detectors for switching the lights off and switching the lights on. **Row 2:** Detectors for pressing a rocker switch. **Row 3:** Detectors for pulling a drawer open and pushing a drawer close.

future work, we hope to address cases where there are multiple adjacent mechanisms with similar effects next to each other such situations where there are multiple light switches next to each other.

ACKNOWLEDGMENTS

We thank Aaron Bobick, Jim Rehg, and Tucker Hermans for their input. We also thank Willow Garage for the PR2 robot and additional support. This work was funded in part by NSF award IIS-0705130.

REFERENCES

[1] Chih-chung Chang and Chih-jen Lin. LIBSVM : a Library for Support Vector Machines. pages 1–39, 2011.

[2] Hao Dang and Peter K Allen. Robot Learning of Everyday Object Manipulations via Human Demonstration. *IROS*, pages 1284–1289, 2010.

[3] A.N. Erkan, Oliver Kroemer, Renaud Detry, Yasemin Altun, Justus Piater, and Jan Peters. Learning probabilistic discriminative models of grasp affordances under limited supervision. In *IROS*, pages 1586–1591. IEEE, 2010.

[4] Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and Gil Jones. Contact-reactive grasping of objects with partial shape information. In *IROS*, 2010.

[5] Advait Jain and Charles C. Kemp. EL-E: an assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots*, 2009.

[6] Dov Katz and Oliver Brock. Manipulating Articulated Objects With Interactive Perception. In *ICRA*, 2008.

[7] Dov Katz, Jacqueline Kenney, and Oliver Brock. How Can Robots Succeed in Unstructured Environments? In *RSS, Robot Manipulation Workshop*, 2008.

[8] Charles C Kemp and Aaron Edsinger. Robot Manipulation of Human Tools : Autonomous Detection and Control of Task Relevant Features. In *ICDL*, 2006.

[9] Ellen Klingbeil, Ashutosh Saxena, and Andrew Y Ng. Learning to Open New Doors. In *RSS Workshop on Robot Manipulation*, 2008.

[10] Jens Kober, E Oztop, and Jan Peters. Reinforcement Learning to adjust Robot Movements to New Situations. In *RSS*, 2010.

[11] Jeremy Maitin-shepard, Marco Cusumano-towner, Jinna Lei, and Pieter Abbeel. Cloth Grasp Point Detection based on Multiple-View Geometric Cues with Application to Robotic Towel Folding. In *ICRA*, 2010.

[12] Hai Nguyen, Advait Jain, Cressel Anderson, and Charles C. Kemp. A clickable world: Behavior selection through pointing and context for mobile manipulation. In *IROS*, 2008.

[13] Peter Pastor, Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, and Stefan Schaal. Skill Learning and Task Outcome Prediction for Manipulation. In *ICRA*, 2011.

[14] Marcos Salganicoff, Lyle H. Ungar, and Ruzena Bajcsy. Active learning for vision-based robot grasping. *Machine Learning*, 1996.

[15] Ashutosh Saxena, J. Driemeyer, and Andrew Y Ng. Robotic Grasping of Novel Objects using Vision. *IJRR*, 2008.

[16] Greg Schohn and David Cohn. Less is More : Active Learning with Support Vector Machines. In *ICML*, 2000.

[17] Vladimir Sukhoy and Alexander Stoytchev. Learning to Detect the Functional Components of Doorbell Buttons Using Active Exploration and Multimodal Correlation. In *IEEE International Conference on Humanoid Robots*, 2010.